doi:10.11887/j.cn.202502004 http://journal.nudt.edu.cn

柔性作业车间调度问题的课程强化学习算法

卢 超[1],肖 洋[1],张 彪[2],高 亮[3*]

(1. 中国地质大学(武汉)计算机学院,湖北 武汉 430078;2. 聊城大学 计算机学院,山东 聊城 252000;
3. 华中科技大学 智能制造装备与技术全国重点实验室,湖北 武汉 430074)

摘 要:针对深度强化学习在柔性作业车间调度问题上泛化能力不足的问题,提出结合课程学习和深度强化学习的方法。通过动态调整训练实例难度,重点增强最难实例的训练,以适应不同数据分布,避免学习过程中的遗忘问题。仿真测试结果表明,算法在未经训练的大规模问题和基准数据集上保持了不错的性能。在2种人造分布的4个未训练大规模问题上取得了更好的性能表现。相较于精确方法和元启发式方法,对于计算量较大的问题实例,能快速地获得质量不错的解。同时算法可以适应不同的数据分布的柔性作业车间调度问题,具有较快收敛速度和较好泛化能力。

关键词:柔性作业车间调度;深度强化学习;课程学习

中图分类号:TH165 **文献标志码**:A **文章编号**:1001-2486(2025)02-049-11

论文拓展

Curriculum reinforcement learning algorithm for flexible job shop scheduling problems

LU Chao[1], *XIAO Yang*[1], *ZHANG Biao*[2], *GAO Liang*[3*]

(1. School of Computer Science, China University of Geosciences(Wuhan), Wuhan 430078, China;
2. School of Computer Science and Technology, Liaocheng University, Liaocheng 252000, China;
3. State Key Laboratory of Intelligent Manufacturing Equipment and Technology, Huazhong University of Science and Technology, Wuhan 430074, China)

Abstract: To address the issue of the lack of generalization capability of deep reinforcement learning in flexible job shop scheduling problems, a method combining curriculum learning and deep reinforcement learning was proposed. The training instance difficulty was dynamically adjusted, with an emphasis on enhancing the training of the most difficult instances, to adapt to different data distributions and avoid the forgetting problem during the learning process. Simulation test results demonstrate that the algorithm maintained decent performance on large-scale untrained problems and benchmark datasets. It achieves better performance on four large-scale untrained problems with two artificial distributions. Compared to exact methods and metaheuristic methods, for problem instances with larger computational complexity, it could rapidly obtain solutions of decent quality. Moreover, the algorithm can adapt to flexible job shop scheduling problems with different data distributions, exhibiting a relatively fast convergence speed and good generalization capability.

Keywords: flexible job shop scheduling; deep reinforcement learning; curriculum learning

作业车间调度问题(job-shop scheduling problem, JSP)是一个经典的组合优化问题,对于实际任务的调度优化具有重大影响。随着新一代人工智能技术的发展,企业的制造、改进和分销模式朝着快速、智能和灵活的制造方向发展[1-3]。柔性作业车间调度问题[4](flexible job-shop scheduling problem, FJSP)是作业车间调度问题的扩展。相较于作业车间调度问题只需要确定作业排序,柔性作业车间调度问题包含2个子问题:确定各工件的加工机器选择子问题和确定各个机器上的加工工序排序子问题。

FJSP是一个NP-hard问题[4],因此使用传统数值优化方法(如约束规划方法)求解其最优解具有挑战性[5]。对它的研究在学术界和工业界

收稿日期:2024-01-04

基金项目:国家自然科学基金资助项目(52175490,51805495,52175490);湖北省重点研发计划资助项目(2022BAD121)

第一作者:卢超(1986—),男,湖北咸宁人,副教授,博士,硕士生导师,E-mail:luchao@cug.edu.cn

***通信作者**:高亮(1974—),男,山东临清人,教授,博士,博士生导师,E-mail:gaoliang@mail.hust.edu.cn

引用格式:卢超,肖洋,张彪,等. 柔性作业车间调度问题的课程强化学习算法[J]. 国防科技大学学报, 2025, 47(2): 49-59.

Citation: LU C, XIAO Y, ZHANG B, et al. Curriculum reinforcement learning algorithm for flexible job shop scheduling problems[J]. Journal of National University of Defense Technology, 2025, 47(2): 49-59.

一直是一个热点,现有的一些研究方法主要分为启发式调度规则[6-7]、元启发式算法[8-9]和深度强化学习(deep reinforcement learning, DRL)算法[10-11]。①启发式调度规则可以灵活地对调度计划中的意外事件做出反应,计算复杂度低且实现简单,从而能实现最佳的时间效率。然而,设计此类启发式规则是一项艰巨的任务,需要对问题有专门的了解,需要一定的先验知识和大量的试错成本。并且其既不能保证局部最优,也不能保证全局最优[12]。②元启发式算法是解决车间调度问题最常用的优化算法,通过不同的优化迭代算子在车间调度问题上搜索到局部最优解。元启发式算法在调度问题上可以获得高于启发式规则方法的准确度,目前此类方法广泛应用在各类车间调度问题上。但是元启发式算法需要多次迭代才能找到最优解,收敛速度较慢。并且由于元启发式算法需要在搜索空间中进行大量的随机尝试,因此计算量较大[13]。③深度强化学习算法是一种通过与环境交互来学习最优行为的机器学习方法[14]。在车间调度问题中,深度强化学习算法可以通过与环境交互来学习最优的调度策略。相比于启发式调度规则和元启发式算法,深度强化学习算法可以通过与环境交互来学习最优行为,因此可以处理车间调度中的不确定性。同时,深度强化学习算法能够快速处理复杂问题(强化学习模型在训练时需要跟环境多次交互产生大量数据来达到收敛,但在应用时可以快速处理车间调度中较复杂的问题)。与元启发式算法一样,深度强化学习算法同样需要大量迭代训练和计算资源。但是元启发式算法对于每一个问题都从头计算,每一个问题都需要一定的计算量和耗时。而深度强化学习算法在训练过程中计算量大和耗时,但在具体应用时可以非常快地给出一个较好的解。深度强化学习算法可以根据环境的反馈来自适应地调整策略,因此可以更好地适应车间调度中的变化。

但是目前大多数现有的深度强化学习算法都只是针对特定规模的问题训练,存在内存利用效率低的问题[10]。虽然在简单的场景中可以对每个特定规模的问题单独训练得到不错的效果,但是在复杂环境中学习成本不断增高,学习时间过长,泛化问题尚未得到解决。因此,现有技术需要一种提高柔性作业车间调度问题的深度强化学习模型收敛速度和泛化能力的方法。课程学习(curriculum learning, CL)是一种机器学习方法,最初的概念是由 Bengio 等提出的[15]。它将训练数据集分为多个难度级别,并按照从易到难的顺序对模型进行训练。这种方法可以有效地解决模型收敛和泛化能力的问题。

针对上述问题,本文提出一种采用课程学习训练策略的深度强化学习算法,用于求解柔性作业车间调度方法。①通过结合图嵌入方法和注意力机制,构建了一种适应不同问题规模的深度强化学习算法。在应用时可以通过一个模型对不同规模问题实例快速给出较好质量的解。②通过类比人类在教育领域的常见做法,设计了一种可以不断返回表现最差的实例规模进行再学习的阶梯课程学习算法,可以解决现有模型泛化能力不足的问题。

1 柔性作业车间调度问题的数学模型

首先描述柔性作业车间调度问题,然后建立模型并定义问题涉及的参数和变量。

1.1 问题描述

柔性作业车间调度问题可以描述为涉及 n 个工件和 m 台机器的调度问题,用集合 $J=\{J_1,J_2,\cdots,J_n\}$ 和 $M=\{M_1,M_2,\cdots,M_m\}$ 分别表示工件集合和机器集合。其中每个工件 J_i 由 n_i 道工序组成,这些工序必须按照特定的顺序处理,由 $O_i=\{O_{i1},O_{i2},\cdots,O_{in_i}\}$ 表示。所有工件的所有工序构成一个集合 $O=\bigcup_{i=1}^{n}O_i$。每个工序 O_{ij} 有多台机器可以选择,但是最终只能选择在可用机器集合 $M_{ij}\subseteq M$ 中的一台机器上处理。每道工序的加工时间与选择的机器有关,工序 O_{ij} 在机器 $M_k\in M_{ij}$ 上的处理时间被表示为 $p_{ij}^k>0$。柔性作业车间调度问题需要为每个工序确定一个适当的加工机器和机器上的排序以获得一个符合调度性能指标的良好调度方案。此外对于柔性车间调度问题还需要满足如下约束:

1)所有工件在初始 0 时刻同时到达;

2)所有机器在初始 0 时刻均空闲,能够被安排执行加工任务;

3)每个工序必须分配给一个合适的机器且只能分配给一个机器;

4)每台机器一次最多可以处理一个工序;

5)机器在加工过程中不可被中断;

6)不考虑机器的准备时间和维修时间;

7)不考虑工件的运输时间。

析取图是描述作业车间调度问题和柔性作业车间调度问题的良好工具。图 1 和图 2 分别展示了一个柔性作业车间调度问题的实例和一个该问题的可行解。图 1 ~2 中不同颜色的线分别表示

不同的机器,S 和 E 分别是虚拟的开始节点和约束节点。图 1 中虚线代表机器可处理的工序关系,图 2 中实线代表已经调度好的工序关系。

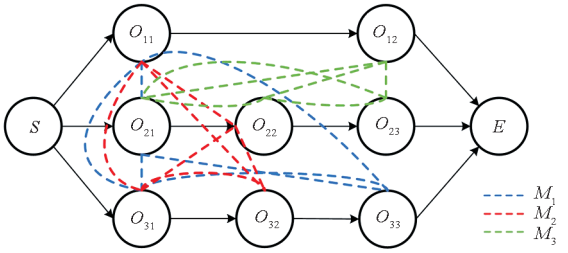

图 1　柔性作业车间调度问题实例

Fig. 1　Instance of flexible job shop scheduling problem

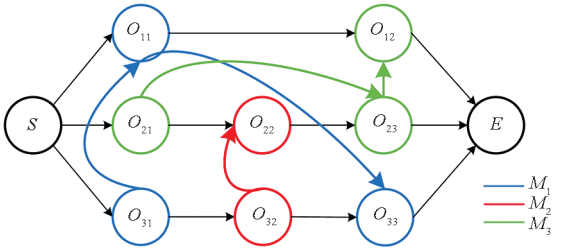

图 2　可行解

Fig. 2　Feasible solution

1.2　模型构建

表 1 为柔性作业车间调度问题的数学模型涉及的符号的说明。

表 1　柔性作业车间问题符号说明

Tab. 1　Notations description of flexible job shop problems

类别	符号	意义
集合	J	工件集合
	M	机器集合
	O	工序集合
参数	p_{ij}^k	工序 O_{ij} 在机器 M_k 上的处理时间
	s_{ij}	工序 O_{ij} 的加工开始时间
	c_{ij}	工序 O_{ij} 的加工完成时间
	C_i	工件 J_i 的完工时间
	$C_{\max}$	最大完工时间
	x_{ijk}	工序 O_{ij} 是否在机器 M_k 上加工
	y_{ijefk}	工序 O_{ij} 是否在机器 M_k 上比工序 O_{ef} 先加工
	n	工件总数
	n_i	工件 J_i 的工序总数
	m	机器总数
索引	i,e	工件序号
	j,f	工序序号
	k	机器序号

本文以最小化最大完工时间 $C_{\max}$ 为优化目标。最大完工时间是从作业最早开始时间到最后结束时间的时间长度,具体为最后一道工序完成的时间,是衡量调度方案的最根本指标,主要体现车间的生产效率。柔性作业车间调度问题的数学模型如下所示。

目标函数:

$$\min\ C_{\max} = \min \max_{1\leqslant i\leqslant n} C_i \tag{1}$$

约束条件:

$$s_{ij} + p_{ij}^k \leqslant c_{ij} \tag{2}$$

$$c_{ij} \leqslant s_{i(j+1)},\ j+1 \leqslant n_i \tag{3}$$

$$c_{in_i} \leqslant C_{\max} \tag{4}$$

$$(s_{ij} + p_{ij}^k) \cdot y_{ijefk} \leqslant s_{ef} \tag{5}$$

$$y_{ijefk} = \begin{cases} 1, & \text{工序 } O_{ij} \text{在机器 } M_k \text{ 上比工序 } O_{ef} \text{先加工} \\ 0, & \text{其他} \end{cases} \tag{6}$$

$$\sum_k^m x_{ijk} = 1 \tag{7}$$

$$x_{ijk} = \begin{cases} 1, & \text{工序 } O_{ij} \text{在机器 } M_k \text{ 上加工} \\ 0, & \text{其他} \end{cases} \tag{8}$$

$$\begin{cases} s_{ij} \geqslant 0 \\ p_{ij}^k \geqslant 0 \\ c_{ij} \geqslant 0 \end{cases} \tag{9}$$

式(1)描述了柔性作业车间调度问题的优化目标函数;式(2)限制了工序必须在机器上不间断地处理;式(3)限制了工件的工序必须按照规定的顺序处理;式(4)表示每个工件的完工时间不能超过总的完工时间;式(5)和式(6)限制了机器在同一时刻只能处理一道工序;式(7)和式(8)限制了每道工序只能安排在一台机器上加工;式(9)要求工序的开始加工时间、处理时间和完成时间都大于或等于 0。

2　算法构建

本文参考了 Wang 等提出的深度强化学习模型[16],并将其训练过程重构,使其能够在不同的大小规模上训练模型,这是本文能够采用课程学习算法进行泛化研究的基础。

2.1　深度强化学习算法

如图 3 所示,经典的强化学习范式包括生成动作的智能体、返回奖励和更新状态的环境。给定一个 $n \times m$ 规模的柔性作业车间调度问题,调度过程如下:其工序总数为 $|O|$,智能体逐步地构造解,在每个决策时刻 t,智能体观察当前系统状态 s_t 并做出动作 a_t,将未调度的工序分配给空闲

机器。环境过渡到下一个决策时刻 $t+1$。该过程不断迭代,直到所有工序都被调度。

图 3 经典强化学习范式

Fig. 3 Classical reinforcement learning paradigm

状态:时刻 t 处所有工序和机器的条件构成状态 s_t。工序可以根据其在确定时刻的状况分成三组:已完成的工序、正在处理的工序和未调度的工序。其中在某时刻 t 的决策只取决于正在处理的工序和未调度的工序,所以称正在处理的工序和未调度的工序为相关工序。已完成的工序不影响后续调度,为不相关工序。在某时刻 t,机器分为两组:不能处理任何剩余未调度工序的机器称为不相关机器,能够处理剩余未调度工序的机器称为相关机器。定义 $O_u(t)$ 和 $M_u(t)$ 为在时刻 t 处的相关工序集合和相关机器集合。在每个状态 s_t,会更新工序 O_{ij} 的开始时间 $s_{ij}(t)$。如果工序 O_{ij} 已经调度,$s_{ij}(t)$ 为确定的值 s_{ij}。如果工序 O_{ij} 未调度,$s_{ij}(t)$ 是一个估计值:如果工序 O_{ij} 的前一个工序 $O_{i(j-1)}$ 已经调度在机器 M_k,则 $s_{ij}(t)=s_{i(j-1)}+p^k_{i(j-1)}$;否则 $s_{ij}(t)=s_{i(j-1)}+\bar{p}_{i(j-1)}$,$\bar{p}_{i(j-1)}=(\sum_{k=1}^{m}p^k_{i(j-1)})/m$ 为工序 $O_{i(j-1)}$ 的平均处理时间。

动作:将工序选择和机器分配结合起来作为一个复合决策。$A(t)$ 为时刻 t 处互相匹配的工序 - 机器对的集合。在某时刻 t 的动作空间(所有可能的动作)是由 $A(t)$ 中所有工序 - 机器对集合定义的。对于一个动作 $a_t\in A(t)$,a_t 是一个可行的工序和机器对 (O_{ij},M_k),O_{ij} 的前一个工序 $O_{i(j-1)}$ 已经调度,$M_k\in M_{ij}$ 是空闲的机器。

状态转换:根据状态 s_t 和动作 a_t,环境会确定性地转移到新的状态 s_{t+1}。

奖励:奖励 $r_t=r(s_t,a_t,s_{t+1})$ 应被设计为引导智能体选择能够最小化整个任务总完成时间的动作。由此奖励定义为 s_t 和 s_{t+1} 完工时间之差,$r_t=C_{\max}(s_t)-C_{\max}(s_{t+1})$。其中 $C_{\max}(s_t)$ 为状态 s_t 时最大完工时间的估计。当折扣因子 $\gamma=1$ 时,累积奖励为 $G=\sum_{t=0}^{|O|}r(s_t,a_t,s_{t+1})=C_{\max}(s_0)-C_{\max}$。对于一个具体的柔性作业车间调度问题,$C_{\max}(s_0)$ 是一个常数,所以根据累计奖励的公式,最小化 $C_{\max}$ 等价于最大化累计奖励 G。真实的完工时间只有在调度完成后才能知道,所以在调度过程中使用不同状态的估计完工时间下界。具体的估计过程如下:$\underline{C}(O_{ij},s_t)=\underline{C}(O_{i(j-1)},s_t)+\min_{k\in M_{ij}}p^k_{ij}$,其中 $\underline{C}(O_{ij},s_t)$ 表示当前工序 O_{ij} 在当前状态 s_t 下的完工时间下界,$\underline{C}(O_{i(j-1)},s_t)$ 表示前序工序 $O_{i(j-1)}$ 在当前状态 s_t 下的完工时间下界。为了计算的一般性,规定 $\underline{C}(O_{i,0})=0$。

策略:随机策略用 $\pi(a_t|s_t)$ 表示,其实质是一个条件概率函数,表示在状态 s_t 对应动作 a_t 的概率分布。$\pi(a_t|s_t)$ 函数可以使用 DRL 算法拟合。给定一个 t 时刻的状态 s_t,该策略函数会返回选择 a_t 的概率分布。

结合嵌入和注意力机制的强化学习如图 4 所示,相关工序 $O_u(t)$ 和相关机器 $M_u(t)$ 嵌入后的向量通过注意力层后与 $A(t)$ 拼接,然后一起作为 actor 和 critic 的输入,因此对于不同规模的问题,都能用同一个模型处理,提高内存利用率。

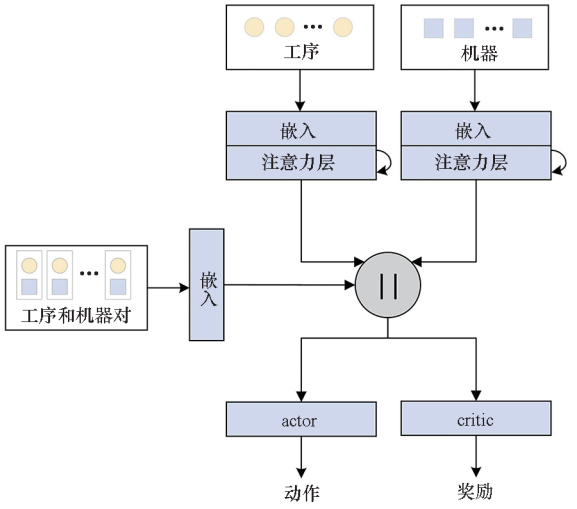

图 4 结合嵌入和注意力机制的强化学习

Fig. 4 Reinforcement learning combined with embedding and attention mechanism

假设 $\boldsymbol{h}^{(0)}_{O_{ij}}$ 和 $\boldsymbol{h}^{(0)}_{M_k}$ 分别为初始未经过注意力层的工序 O_{ij} 和机器 M_k 的原始特征向量。其中工序对应的注意力层模块将相关工序 $O_{ij}\in O_u(t)$ 的特征向量 $\boldsymbol{h}^{(0)}_{O_{ij}}$ 作为输入,通过计算与它的前继工序和后继工序(如果存在)的注意系数,来构建与它的前继工序和后继工序(如果存在)之间的关系。具体的注意系数为:

$$e_{i,j,p}=L(\boldsymbol{a}^{\mathrm{T}}[(\boldsymbol{W}\boldsymbol{h}^{(0)}_{O_{ij}})\,\|\,(\boldsymbol{W}\boldsymbol{h}^{(0)}_{O_{ip}})]) \quad (10)$$

式中:$L(\cdot)$ 是激活函数;$\boldsymbol{a}^{\mathrm{T}}$ 是权重向量,$\boldsymbol{W}$ 是线

性变换矩阵;p 是前继或后继下标,需满足 $|p-j|\leqslant 1$。

通过 softmax 激活函数 $\rho(\cdot)$ 对注意系数 $e_{i,j,p}$ 进行处理,得到归一化后的注意系数:

$$\alpha_{i,j,p}=\rho(e_{i,j,p}) \tag{11}$$

通过注意系数 $\alpha_{i,j,p}$ 变换工序的特征向量:

$$\boldsymbol{h}_{O_{ij}}^{(1)}=\sigma\Big(\sum_{p=j-1}^{j+1}\alpha_{i,j,p}\boldsymbol{W}\boldsymbol{h}_{O_{ip}}^{(0)}\Big) \tag{12}$$

式中,σ 是非线性激活函数。

类似地,机器对应的注意力层模块将相关机器 $M_k\in M_{\mathrm{u}}(t)$ 的特征向量 $\boldsymbol{h}_{M_k}^{(0)}$ 作为输入。计算它与它的所有竞争机器的注意系数 u_{kq}。

显然两台机器在它们都可以处理的工序上存在竞争关系,而随着不断有工序在被调度,这种竞争关系会不断变化。

定义 C_{kq} 为机器 M_k 和机器 M_q 相互竞争的工序集合。定义 $N_k=\{q\,|\,C_{kq}\neq\varnothing\}$ 表示与机器 M_k 存在竞争关系的机器集合。为了方便计算,定义 $\boldsymbol{c}_{kq}=\sum_{O_{ij}\in C_{kq}\cap O_{\mathrm{u}}(t)}\boldsymbol{h}_{O_{ij}}$ 表示机器 M_k 和机器 M_q 之间竞争强度的度量(工序 O_{ij} 的特征向量 $\boldsymbol{h}_{O_{ij}}$ 越大,工序 O_{ij} 越重要,竞争也就越激烈)。

借助 $\boldsymbol{c}_{kq}$ 计算注意系数 u_{kq}:

$$u_{kq}=L(\boldsymbol{b}^{\mathrm{T}}[(\boldsymbol{Z}^1\boldsymbol{h}_{M_k})\,\|\,(\boldsymbol{Z}^1\boldsymbol{h}_{M_q})\,\|\,(\boldsymbol{Z}^2\boldsymbol{c}_{kq})]) \tag{13}$$

式中:$M_k\in M_{\mathrm{u}}(t)$,对每个输入特征 $\boldsymbol{h}_{M_k}$ 计算它与所有竞争机器的 $q\in N_k$ 的注意系数 u_{kq};$\boldsymbol{Z}^1$ 和 $\boldsymbol{Z}^2$ 是权值矩阵;$\boldsymbol{b}^{\mathrm{T}}$ 是线性变换。

值得注意的是,C_{kk} 是机器 M_k 可以处理的未调度工序集合,k 总是属于集合 N_k。因此 $\boldsymbol{c}_{kk}$ 可以被看成机器 M_k 处理能力的一种度量。当集合 $C_{kq}\cap O_{\mathrm{u}}(t)$ 为空时,$\boldsymbol{c}_{kq}$ 置为全零。

通过 softmax 激活函数对注意系数 u_{kq} 进行处理,得到归一化后的注意系数:

$$\alpha_{kq}=\rho(u_{kq}) \tag{14}$$

通过注意系数 α_{kq} 变换机器的特征向量:

$$\boldsymbol{h}_{M_k}^{(1)}=\sigma\Big(\sum_{q\in N_k}\alpha_{kq}\boldsymbol{W}\boldsymbol{h}_{M_q}^{(0)}\Big) \tag{15}$$

假设 $\boldsymbol{h}_{O_{ij}}^{(L)}$ 和 $\boldsymbol{h}_{M_k}^{(L)}$ 分别是 $\boldsymbol{h}_{O_{ij}}^{(0)}$ 和 $\boldsymbol{h}_{M_k}^{(0)}$ 经过 L 层注意力层后获得的特征向量。通过平均池化和拼接操作获得全局特征向量 $\boldsymbol{h}_G^{(L)}$,并作为后面决策部分的输入:

$$\boldsymbol{h}_G^{(L)}=\Big[\Big(\frac{1}{|O_{\mathrm{u}}|}\sum_{O_{ij}\in O_{\mathrm{u}}}\boldsymbol{h}_{O_{ij}}^{(L)}\Big)\Big\|\Big(\frac{1}{|M_{\mathrm{u}}|}\sum_{M_k\in M_{\mathrm{u}}}\boldsymbol{h}_{M_k}^{(L)}\Big)\Big] \tag{16}$$

在决策部分,采用基于 actor-critic 的决策网络。对于 actor 和 critic 分别采用两个多层感知机(multilayer perceptrons, MLPs),它们的参数分别用 θ 和 φ 表示。

具体地,actor 网络通过两个步骤生成随机策略函数 $\pi_\theta(a_t\,|\,s_t)$:

首先通过连接所有与动作 $a_t=(O_{ij},M_k)$ 有关的特征向量作为多层感知机 C_θ 的输入得到动作的标量函数 $\mu(a_t\,|\,s_t)$:

$$\mu(a_t\,|\,s_t)=C_\theta[\boldsymbol{h}_{O_{ij}}^{(L)}\,\|\,\boldsymbol{h}_{M_k}^{(L)}\,\|\,\boldsymbol{h}_G^{(L)}\,\|\,\boldsymbol{h}_{(O_{ij},M_k)}] \tag{17}$$

然后通过 softmax 把动作的标量函数 $\mu(a_t\,|\,s_t)$ 转化成策略函数 $\pi_\theta(a_t\,|\,s_t)$:

$$\pi_\theta(a_t\,|\,s_t)=\frac{\exp(\mu(a_t\,|\,s_t))}{\sum_{b_t\in A(t)}\exp(\mu(b_t\,|\,s_t))} \tag{18}$$

式中,$\pi_\theta(a_t\,|\,s_t)$ 输出的是在状态 s_t 下选择动作 a_t 的概率。

类似地,critic 网络以 $\boldsymbol{h}_G^{(L)}$ 作为多层感知机 C_φ 的输入,生成一个标量函数 $\nu_\varphi(s_t)$ 作为状态 s_t 价值的估计。

$$\nu_\varphi(s_t)=C_\varphi[\boldsymbol{h}_G^{(L)}] \tag{19}$$

2.2　课程学习算法

为了解决模型的泛化问题,引入如图 5 所示的课程学习算法。Iklassov 等曾利用课程学习提高了深度强化学习在 JSP 问题上的泛化能力[17]。本文针对柔性车间调度问题设计了增强自适应阶梯课程学习,以提高模型的泛化能力。课程学习是一种模仿人类课程中有意义的学习顺序,从简单的数据向困难的数据训练机器学习模型的训练策略。课程学习策略作为一种易于使用的插件,在计算机视觉和自然语言处理等场景中,能提高各种模型的泛化能力和收敛速度。

如图 6 所示,课程学习算法是一种使用渐进式复杂的数据进行策略学习的训练方法。简而言之,课程学习意味着“从容易的数据训练到难的数据”。更具体地说,其基本思想是“从小处开始”,用更容易的数据子集(或更容易的子任务)

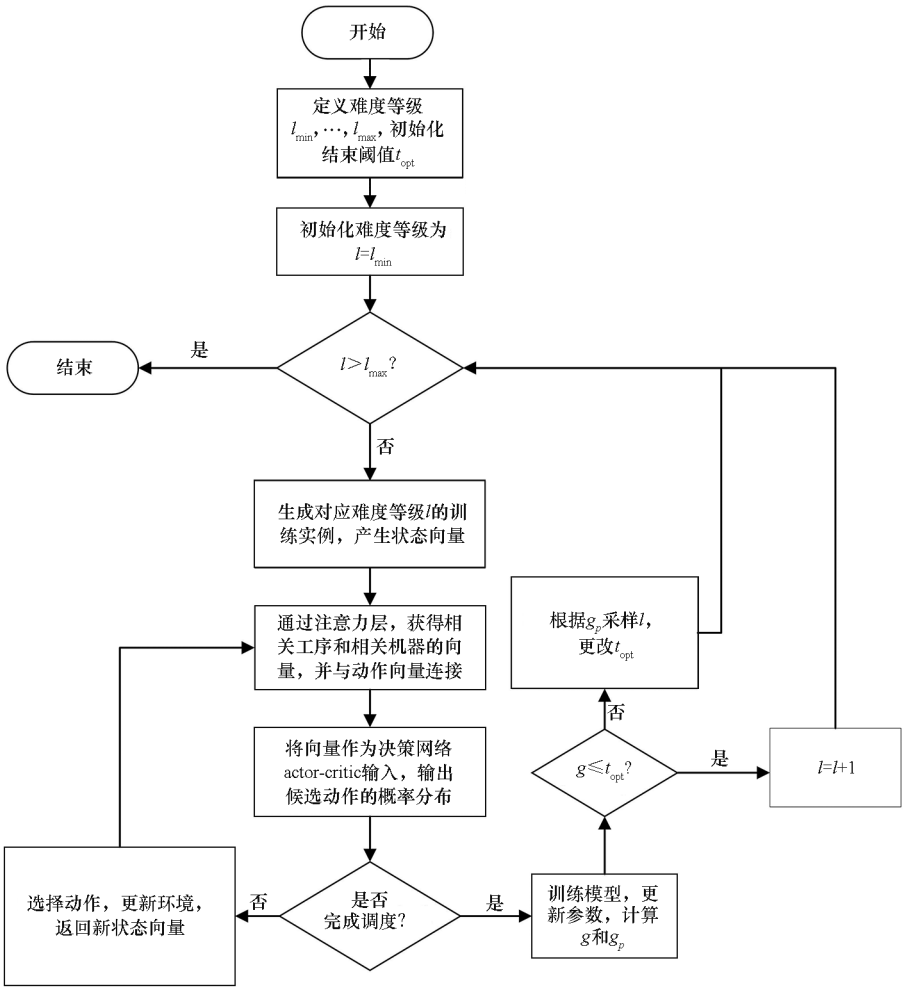

图 5 课程学习训练流程

Fig. 5 Curriculum learning training flow

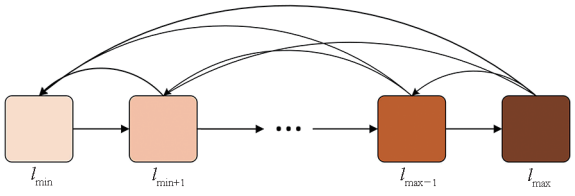

图 6 课程学习难度等级图

Fig. 6 Curriculum learning difficulty level chart

训练机器学习模型，然后逐渐提高数据（或子任务）的难度水平，直到训练整个数据集（或目标任务）。

如算法 1 所示，将不同规模的实例划分成不同的难度等级，训练从最低难度等级 $l=l_{min}$ 开始，目的是达到最高难度等级 $l=l_{max}$。传统的课程学习往往在求解大规模问题时会出现灾难性遗忘的问题。为了解决这个问题，提出了增强自适应阶梯课程学习（enhanced adaptive staircase curriculum learning，EASCL）算法。具体地，每 i 次迭代后将预期结果和最优解的结果进行比较得到差值 g。在数值上，算法对每个难度等级 $l' \leqslant l$ 的 g 进行归一化得到 $g(l)$ 数组，并将其转化为概率形式。这样 g 越大就会被赋予越高的概率，而每个概率对应一个特定的问题大小。智能体在难度等级 l 迭代学习 u 次后，将 $g(l)$ 与阈值 t_{opt} 进行比较，当 $g(l) \leqslant t_{opt}$ 时，EASCL 选择下一个难度等级 $l \leftarrow l+1$。否则，算法从 g 的分布中选取一个更小的困难等级 l'。该算法通过重新访问已学习过的难度等级和锚定最困难的难度等级来强化训练，这极大避免了遗忘问题。

3 仿真实验

在本节中，将本文算法与几种基准方法进行比较，包括几种流行的启发式规则、Google OR-Tools 的精确方法；并且将本文算法与当前先进的两种基于 DRL 的方法[16, 18]进行对比。

3.1 数据集

一个有 n 个作业和 m 台机器的 FJSP 实例简称为“$n \times m$”（工序的数量在不同的实例中有所

算法 1　增强自适应阶梯课程学习

Alg. 1　Enhanced adaptive staircase curriculum learning

输入:训练数据 *train_data*,测试数据 *test_data*,非负的浮动最优阈 t_{opt},正整数 u、正实数 λ、$failCount=0$ 和 $l_{\max}$,初始化网络参数 $\theta \leftarrow \theta_0$ 和非负整数 $l \leftarrow l_0$

while $l \leqslant l_{\max}$ **do**
  for $i=0,1,\cdots$ **do**
    使用 *test_data* 的最优解计算 $g(l)$ 数组
    根据 $g(l)$ 数组归一化为 g_p 概率分布
    if $i\%u==0$ **then**
      if $g(l) \leqslant t_{\text{opt}}$ **then**
        更新 $l \leftarrow l+1$
      else
        $failCount \leftarrow failCount+1$
        从 g_p 概率分布采样难度等级 l
        $t_{\text{opt}} \leftarrow t_{\text{opt}}+0.01\times e^{\lambda \times failCount}$
      end if
    end if
    根据 *train_data* 更新 θ
  end for
end while
return θ

不同)。与大多数相关工作一样,本文采用人工合成数据(synthetic data, SD)的 FJSP 实例用于训练和测试。第一种数据改编自文献[18],它允许作业拥有不同数量的工序,用 SD1 表示,其生成方法与文献[19]中众所周知的过程类似。如表 2 所示,对于 SD1 生成的 $n\times m$ 的 FJSP 实例,每个作业 J_i 有 n_i 个工序,n_i 是从均匀分布 $U(0.8m, 1.2m)$ 中采样的整数。对于每个工序 O_{ij},可以选择的机器集合 $|M_{ij}|$ 和处理时间 p_{ij}^k 是从均匀分布 $U(1,m)$ 和 $U(1,20)$ 采样的整数。

表 2　数据集 SD1 实例分布

Tab. 2　Dataset SD1 instance distributions

$n\times m$	n_i	$\lvert M_{ij}\rvert$	p_{ij}^k
10×5	$U(4,6)$	$U(1,5)$	$U(1,20)$
20×5	$U(4,6)$	$U(1,5)$	$U(1,20)$
15×10	$U(8,12)$	$U(1,10)$	$U(1,20)$
20×10	$U(8,12)$	$U(1,10)$	$U(1,20)$
30×10	$U(8,12)$	$U(1,10)$	$U(1,20)$
40×10	$U(8,12)$	$U(1,10)$	$U(1,20)$

第二种数据改编自文献[16],其工序的随机处理时间范围更大,用 SD2 表示。如表 3 所示,对于 SD2 生成的 $n\times m$ 的 FJSP 实例,每个作业有 n_i 个工序。对于每个工序 O_{ij},可以选择的机器集合 $|M_{ij}|$ 和处理时间 p_{ij}^k 是从均匀分布 $U(1,m)$ 和 $U(1,99)$ 采样的整数。

表 3　数据集 SD2 实例分布

Tab. 3　Dataset SD2 instance distributions

$n\times m$	n_i	$\lvert M_{ij}\rvert$	p_{ij}^k
10×5	5	$U(1,5)$	$U(1,99)$
20×5	5	$U(1,5)$	$U(1,99)$
15×10	10	$U(1,10)$	$U(1,99)$
20×10	10	$U(1,10)$	$U(1,99)$
30×10	10	$U(1,10)$	$U(1,99)$
40×10	10	$U(1,10)$	$U(1,99)$

为了方便比较,考虑了 6 种不同的 FJSP 问题规模:10×5、20×5、15×10、20×10、30×10、40×10。在 4 个较小的规模上执行训练,并使用最大的 2 种规模(30×10 和 40×10)来测试训练策略的泛化能力。测试数据是预先生成的,每种规模各有 100 个实例。

同时为了更好地比较,本文和文献[16]一样在 4 组公共基准数据集上评估训练后的模型,以探索它们在交叉分布任务上的能力,包括文献[18]中的 mk1 – 10 实例和文献[20]中的 3 组 la1 – 40 实例。

3.2　参数配置

本文算法在 Pycharm(Community Edition)中基于 Python 3.10 语言编程实现,使用 PyTorch 和 Gym 框架,实验平台为 Intel © Core™ i7 – 12650H (@2.30GHz) CPU、NVIDIA GeForce RTX4060 和 8 GB RAM,运行在 Windows11 64 bit 系统。为了方便比较,参数设置与文献[16]和文献[18]中的一致。σ 激活函数采用指数线性单元(exponential linear unit, ELU)。C_θ 和 C_φ 都有两个隐藏层,维度为 64,tanh 作为激活函数。在损失函数中设置策略、值和熵系数分别为 1、0.5 和 0.01。剪切参数、广义优势估计(generalized advantage estimation, GAE)参数和折扣因子分别设置为 0.2、0.98 和 1。采用 Adam 优化器,学习率 $r=0.000\,3$。

3.3　基准方法和性能指标

对于基准方法,本文选择了 4 个在文献[21]中显示出良好性能的启发式规则(priority

dispatching rules, PDRs), 并将其推广到求解FJSP。在实验中,考虑到随机性质,每个启发式规则的结果采取5次独立运行的平均结果。具体启发式规则如下:

先进先出(first-in-first-out, FIFO):选择最早就绪的候选工序和最早就绪的可执行机器。

剩余最多工序(maximum operation processing number remaining, MOPNR):选择具有最多剩余后续工序的工序和可以立即处理它的机器。

最短加工时间(shortest processing time, SPT):选择加工时间最短的工序和机器的兼容对。

剩余最多工作量(most work remaining, MWKR):选择剩余后续工序的平均处理时间最多的工序以及能够立即处理该工序的机器。

评估指标使用平均的最大完工时间(Obj.)和最优解(最优近似解)之间的差距值(Gap),两者都是数值越小越好。为了与最优解进行比较,参考了基于约束规划的精确求解器Google OR-Tools,并且以其结果作为参考线计算Gap。Gap计算过程为:(当前解完工时间 - 最优解完工时间)/最优解的完工时间 × 100%。为了便于比较,直接导入文献[16]和文献[18]的公开结果,文献[18]为未使用注意力机制的深度强化学习方法,文献[16]为使用注意力机制的深度强化学习方法。包括在所有6种规模的测试实例的结果和在公共基准数据集上的结果,文献[16]和文献[18]都为一种规模对应一个模型,本方法为一个通用模型。在测试中和文献[18]一样采取贪婪和采样策略两种选择动作的策略。具体来说,采样策略根据actor网络的输出概率分布并行地采样一个实例100次,并记录最好的一次,以在可接受的计算负担下提高解决方案的质量。贪婪策略根据actor网络的输出概率分布每次贪婪地选取概率最大的动作。

模型的性能根据平均的最大完工时间和其与最优解(最优近似解)的相对差距进行评估。最优解(最优近似解)要么是OR-Tools对实例的解,要么是文献[22]中公开的公共基准数据集的最佳结果。在实验中,分别比较了贪婪和采样策略两种策略的性能,并在每个问题中用粗体强调了最佳结果。

3.4 结果

3.4.1 中等规模合成数据集上的结果

表4比较了本文模型和4种流行的启发式规则在测试实例上的Obj.和Gap。这些测试实例是使用与训练实例相同的分布生成的,包含了SD1和SD2两种分布和10×5、20×5、15×10、20×10四种规模。同时比较了本模型和文献[16]、文献[18]提出的方法,文献[16]和文献[18]的结果是公开的,为了公平比较,采用了相同的测试实例。对于文献[16]、文献[18]的模型,相同规模问题采用相同规模的模型进行比较。

从粗体数字中可以清楚地看出,对于所有问题规模的两个合成数据,本模型不仅明显优于所有启发式规则,而且相对于文献[16]和文献[18]中的DRL解决方案,也表现出了较好的改进。

4种启发式规则受数据分布的影响较大,对于同样规模的问题,在SD2的数据集上表现出大幅度的降低,而DRL方法普遍保持了良好的性能表现。

可以看出,采取采样策略的结果明显好于采取贪婪策略的结果。在2种不同分布的6种规模的测试实例中,采样策略在6个结果上比贪婪策略提升5%。在所有的12种结果中采样策略均好于贪婪策略,平均提升6%。

另外,本模型在SD1数据上的20×10规模的测试实例上以1.05%的优势超越了OR-Tools的结果。可以看出,所有方法在SD2数据集上的表现都不如在同等规模的SD1数据集上的表现,当处理时间范围变大时,性能会受到影响,导致与最佳解决方案的差距增大。在这种情况下,本方法仍然表现良好,特别是在使用采样策略时,在规模为15×10和20×10的SD2数据集的测试实例采样策略相对于贪婪策略分别提升了17%和12%。

3.4.2 泛化能力分析

如图7所示,在6种规模的合成数据集上,本模型对比了表现最好的文献[16]中的4个模型。除了在较小的问题规模(10×5和15×10)上表现略差,本模型在其余问题规模均达到了较好的表现。

表 4　在中等规模问题上的结果

Tab. 4　Results on medium-sized problems

数据集	问题规模	指标	PDRs				贪婪策略			采样策略		
			FIFO	MOPNR	SPT	MWKR	文献[18]	文献[16]	本文	文献[18]	文献[16]	本文
SD1	10×5	Obj.	119.40	115.38	129.82	113.23	111.67	106.71	**106.56**	105.59	101.67	**100.78**
		Gap	24.06%	19.87%	34.76%	17.58%	16.03%	10.87%	**10.65%**	9.66%	5.57%	**4.63%**
	20×5	Obj.	216.08	214.16	230.48	209.78	211.22	197.56	**195.72**	207.53	192.78	**192.01**
		Gap	14.87%	13.85%	22.56%	11.51%	12.27%	5.03%	**4.04%**	10.31%	2.46%	**2.06%**
	15×10	Obj.	184.55	173.15	198.33	171.25	166.92	161.28	**160.14**	160.86	153.22	**151.24**
		Gap	28.65%	20.68%	38.22%	19.41%	16.33%	12.42%	**11.57%**	12.13%	6.79%	**5.4%**
	20×10	Obj.	233.48	219.80	255.17	216.11	215.78	**198.50**	198.52	214.81	193.91	**193.88**
		Gap	19.22%	12.20%	30.25%	10.30%	10.15%	**1.31%**	**1.31%**	9.64%	−1.03%	**−1.05%**
SD2	10×5	Obj.	569.41	557.48	514.39	549.28	553.61	408.40	**407.99**	483.90	366.74	**364.84**
		Gap	76.47%	72.52%	57.96%	70.01 %	71.42%	25.68%	**25.41%**	49.71%	12.57%	**11.95%**
	20×5	Obj.	1 045.83	1 045.94	835.94	1 026.03	1 059.04	671.03	**663.15**	962.90	629.94	**624.04**
		Gap	74. 59%	74.58%	38.91%	71.31%	76.79%	11.52%	**10.24%**	60.70%	4.66%	**3.67%**
	15×10	Obj.	871.14	845.16	703.07	830.53	807.47	591.21	**583.4**	756.07	521.83	**517.37**
		Gap	132.23%	125.32%	86.74%	121.45%	115.26%	57.16%	**55.17%**	101.52%	38.70%	**37.5%**
	20×10	Obj.	1 088.05	1 059.68	829.14	1 040.69	1 045.82	610.16	**605.63**	990.37	552.64	**547.05**
		Gap	135.27%	129.09%	78.82%	124.98%	126.12%	31.58%	**30.64%**	114.15%	19.13%	**17.97%**

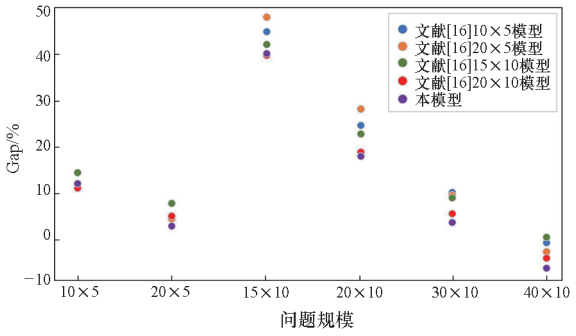

图 7　不同问题规模 Gap 对比

Fig. 7　Gap comparison for different problem scales

如表 5 所示,在未训练过的规模为 30×10 和 40×10 的测试实例中比较模型在 2 个数据集上的泛化性能。由于文献[16]和文献[18]中一个问题规模训练一个模型,选取其表现最好的 20×10 模型用于比较。可以看出,除了在 SD1 数据集的贪婪策略上的结果略差于文献[16]的结果,本模型在计算成本可接受的情况下,始终在未训练过的大规模数据上表现良好。在使用采样策略时本模型优于其他所有比较的方法。值得注意的是,在规模为 40×10 的 SD2 数据实例上,本模型以 7.09% 的优势超越了 OR-Tools 的结果。

这些结果表明本模型采用的增强自适应阶梯课程学习算法可以通过小规模任务的训练来学习一般知识,这些知识可以用于解决看不见的大规模实例。同时相较于文献[16]和文献[18]中的方法,本模型具有较好的泛化能力。

由于真实世界的问题可能来自未知的分布。因此如表 6 所示,进一步测试了本模型(在合成数据上训练)在 4 个公共基准数据集上的性能。每个基准数据集包含不同问题规模的实例。

对于文献[16]和文献[18],选择了在这些基准测试中取得了最好的结果的模型比较(在 SD1 数据上训练的规模为 10×5 和 15×10 的模型)。对于启发式规则,在表 6 中只列出了 4 个 PDR 中表现最好的 MWKR 的结果。

在 4 个公共基准数据集上,可以看出本模型取得了良好的性能。在大多数情况下超过文献[16]和文献[18],在其余情况下表现相当。分别在 mk 实例和 la(vdata)实例的采样策略、la(rdata)实例和 la(vdata)实例的贪婪策略取得了最好结果。这些结果展示了课程学习能够确实提升模型的泛化能力,能够真正捕捉到 FJSP 问题的内在结构信息,适应不同分布和不同规模的问题,而不仅仅是学习特定规模问题的知识。

表5 在大规模问题上的结果

Tab. 5 Results on large-scale problems

数据集	问题规模	指标	Top PDRs		贪婪策略			采样策略		
			SPT	MWKR	文献[18] 20×10	文献[16] 20×10	本文	文献[18] 20×10	文献[16] 20×10	本文
SD1	30×10	Obj.	350.07	312.93	313.04	**281.49**	282.49	312.59	279.20	**278.99**
		Gap	27.47%	13.96%	14.01%	**2.50%**	2.87%	13.49%	1.67%	**1.6%**
	40×10	Obj.	445.17	414.82	416.18	**371.45**	371.89	415.25	370.48	**368.73**
		Gap	21.66%	13.37%	13.75%	**1.52%**	1.64%	13.49%	1.14%	**0.77%**
SD2	30×10	Obj.	1 105.99	1 539.67	1 543.69	774.56	**764.74**	1 461.16	725.27	**721.05**
		Gap	59.74%	122.89%	123.57%	11.95%	**10.54%**	111.51%	4.80%	**4.18%**
	40×10	Obj.	1 357.16	2 037.65	2 032.54	962.58	**948.03**	1 945.53	914.02	**909**
		Gap	38.74%	108.66%	108.12%	−1.67%	**−3.11%**	99.26%	−6.60%	**−7.09%**

表6 在基准数据集上的结果

Tab. 6 Results on benchmark data set

基准数据集	指标	Top PDRs	贪婪策略					采样策略				
		MWKR	文献[18] 10×5	文献[18] 15×10	文献[16] 10×5	文献[16] 15×10	本文	文献[18] 10×5	文献[18] 15×10	文献[16] 10×5	文献[16] 15×10	本文
mk	Obj.	201.74	201.40	198.5	185.7	**184.4**	185.1	190.3	190.6	180.8	180.9	**179.5**
	Gap	28.91%	28.52%	26.77%	13.58%	**12.97%**	14.43%	18.56%	19.00%	9.53%	8.95%	**7.89%**
la(rdata)	Obj.	1 053.10	1 030.83	1 031.33	1 031.63	1 040.05	**1 027.17**	985.30	988.38	**978.28**	983.33	982.875
	Gap	13.86%	11.15%	11.14 %	11.42%	12.07%	**10.7%**	5.57%	5.95%	**4.95%**	5.49%	5.56%
la(edata)	Obj.	1 219.01	1 187.48	1 182.08	1 194.98	**1 175.53**	1 188.27	**1 116.68**	1 119.43	1 122.60	1 119.73	1 137.85
	Gap	18.6%	15.53%	15.0%	16.33%	**14.41%**	15.58%	**8.17%**	8.69%	9.08%	8.72%	10.74%
la(vdata)	Obj.	952.01	955.90	954.33	944.85	948.73	**941.8**	930.80	931.33	925. 40	925.68	**924.62**
	Gap	4.22%	4.25%	4.02%	3.28%	3.75%	**2.89%**	1.32%	1.34%	0.69%	0.72%	**0.62%**

4 结论

针对柔性作业车间调度问题,本文提出了一种创新的结合课程学习的深度强化学习端到端框架。该框架整合了注意力机制和深度强化学习技术,并展现出对各种问题规模的良好适应性,允许从小规模问题训练起步,进而成功部署于大规模问题。为了优化这一过程,借鉴教育领域课程学习理念,设计了一种能够针对性地回溯并重新学习性能瓶颈实例规模的算法。实验证明,相较于传统 PDRs 方法及其他现有先进 DRL 方法,本方法在合成数据集和公开基准数据集上展现出了良好性能。特别是在未经直接训练过的不同规模实例上,该模型呈现良好的泛化能力,验证了课程学习策略的有效性。此外,区别于以往针对每个问题规模训练独立模型的做法,本研究提出的单一通用模型在限定内存条件下,能有效应对多种问题规模并保持性能。

然而,面对具有随机不确定性和动态变化属性的 FJSP 问题,以及其他局部可观测的复杂场景时,尽管本方法初步显示出优势,但仍需深入探究如何进一步提升模型在处理此类挑战性问题时的稳健性和泛化能力,这将是未来工作的主要发展方向。

参考文献(References)

[1] ZHANG J, DING G F, ZOU Y S, et al. Review of job shop scheduling research and its new perspectives under Industry 4.0[J]. Journal of Intelligent Manufacturing, 2019, 30: 1809 – 1830.

[2] CHEN B M. On the trends of autonomous unmanned systems research[J]. Engineering, 2022, 12: 20 – 23.

[3] 张洁, 高亮, 李新宇, 等. 前言——工业大数据与工业智能[J]. 中国科学: 技术科学, 2023, 53(7): 1015.
ZHANG J, GAO L, LI X Y, et al. Preface: industrial big data and industrial artificial intelligence[J]. Scientia Sinica Technologica, 2023, 53(7): 1015. (in Chinese)

[4] DAUZÈRE-PÉRÈS S, DING J W, SHEN L J, et al. The flexible job shop scheduling problem: a review[J]. European Journal of Operational Research, 2024, 314(2): 409 - 432.

[5] MENG L L, ZHANG C Y, REN Y P, et al. Mixed-integer linear programming and constraint programming formulations for solving distributed flexible job shop scheduling problem[J]. Computers & Industrial Engineering, 2020, 142: 106347.

[6] HUANG P Y. A comparative study of priority dispatching rules in a hybrid assembly/job shop[J]. International Journal of Production Research, 1984, 22(3): 375 - 387.

[7] 陈亮, 王世进, 周炳海. 柔性作业车间调度问题的集成启发式算法[J]. 计算机工程, 2008, 34(1): 256 - 258.
CHEN L, WANG S J, ZHOU B H. Integrated heuristic algorithm for flexible job-shop scheduling problems[J]. Computer Engineering, 2008, 34(1): 256 - 258. (in Chinese)

[8] YU F, LU C, ZHOU J J, et al. A knowledge-guided bi-population evolutionary algorithm for energy-efficient scheduling of distributed flexible job shop problem[J]. Engineering Applications of Artificial Intelligence, 2024, 128: 107458.

[9] YU F, LU C, ZHOU J J, et al. Mathematical model and knowledge-based iterated greedy algorithm for distributed assembly hybrid flow shop scheduling problem with dual-resource constraints[J]. Expert Systems with Applications, 2024, 239: 122434.

[10] WANG L, PAN Z X, WANG J J. A review of reinforcement learning based intelligent optimization for manufacturing scheduling[J]. Complex System Modeling and Simulation, 2021, 1(4): 257 - 270.

[11] 崔雪艳, 万烂军, 赵昊鑫, 等. 基于深度强化学习的柔性作业车间调度方法[J]. 制造技术与机床, 2023(12): 165 - 170.
CUI X Y, WAN L J, ZHAO H X, et al. A flexible job shop scheduling method based on deep reinforcement learning[J]. Manufacturing Technology & Machine Tool, 2023(12): 165 - 170. (in Chinese)

[12] NEWELL A, SIMON H A. Computer science as empirical inquiry: symbols and search[M]//Association for Computing Machinery, ACM Turing Award Lectures. New York: ACM, 2007: 1975.

[13] 尚正阳, 顾寄南, 唐仕喜, 等. 针对几种元启发式算法的应用性能对比研究[J]. 机械设计与制造, 2021(4): 34 - 38.
SHANG Z Y, GU J N, TANG S X, et al. Comparative study on application performance of several meta-heuristic algorithms[J]. Machinery Design & Manufacture, 2021(4): 34 - 38. (in Chinese)

[14] 王冲, 景宁, 李军, 等. 一种基于多 Agent 强化学习的多星协同任务规划算法[J]. 国防科技大学学报, 2011, 33(1): 53 - 58.
WANG C, JING N, LI J, et al. An algorithm of cooperative multiple satellites mission planning based on multi-agent reinforcement learning[J]. Journal of National University of Defense Technology, 2011, 33(1): 53 - 58. (in Chinese)

[15] BENGIO Y, LOURADOUR J, COLLOBERT R, et al. Curriculum learning[C]//Proceedings of the 26th International Conference on Machine Learning, 2009.

[16] WANG R Q, WANG G, SUN J, et al. Flexible job shop scheduling via dual attention network based reinforcement learning[EB/OL]. (2023 - 06 - 17)[2023 - 11 - 12]. https://arxiv.org/abs/2305.05119v2.

[17] IKLASSOV Z, MEDVEDEV D, SOLOZABAL R, et al. Learning to generalize dispatching rules on the job shop scheduling[EB/OL]. (2022 - 11 - 15)[2023 - 11 - 12]. https://arxiv.org/abs/2206.04423v2.

[18] SONG W, CHEN X Y, LI Q Q, et al. Flexible job-shop scheduling via graph neural network and deep reinforcement learning[J]. IEEE Transactions on Industrial Informatics, 2023, 19(2): 1600 - 1610.

[19] BRANDIMARTE P. Routing and scheduling in a flexible job shop by tabu search[J]. Annals of Operations Research, 1993, 41: 157 - 183.

[20] HURINK J, JURISCH B, THOLE M. Tabu search for the job-shop scheduling problem with multi-purpose machines[J]. Operations-Research-Spektrum, 1994, 15: 205 - 215.

[21] SELS V, GHEYSEN N, VANHOUCKE M. A comparison of priority rules for the job shop scheduling problem under different flow time- and tardiness-related objective functions[J]. International Journal of Production Research, 2012, 50(15): 4255 - 4270.

[22] BEHNKE D, GEIGER M. Test instances for the flexible job shop scheduling problem with work centers[R]. Hamburg: Helmut Schmidt University, 2012.